This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problems Mailbox.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 310 220 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 88306590.6

(51) Int. Cl.4: H03M 13/00

(22) Date of filing: 19.07.88

(30) Priority: 29.09.87 US 102519

(43) Date of publication of application: 05.04.89 Bulletin 89/14

(84) Designated Contracting States: DE FR GB

(71) Applicant: Hewlett-Packard Company
3000 Hanover Street
Palo Alto California 94304(US)

(72) Inventor: Tan, Charles M.C.
21 Del Prado Drive
Campbell California 95008(US)

(74) Representative: Colgan, Stephen James et al
CARPMAELS & RANSFORD 43 Bloomsbury Square
London WC1A 2RA(GB)

(54) An apparatus useful for correction of single bit errors and detection of double bit errors in the transmission of data.

(57) An apparatus which corrects single bit errors and detects double bit errors is presented. The present invention includes an encoder/decoder. During encoding the encoder/decoder produces a plurality of parity bits. These parity bits allow the correction of single bit errors in transmission and the detection of double bit errors in transmission. The encoder/decoder comprises a plurality of arrays. Each array consists of logic gates arranged in the form of a binary tree. Each array generates a single parity bit. Further every array receives as input the same number of data bits; however, each array receives a unique combination of data bits.

When the encoder/decoder is used in decoding, the encoder/decoder produces a plurality of syndrome bits. These syndrome bits are used to correct single bit errors in transmission and to detect double bit errors in transmission. When decoding, each array of logic gates within the encoder/decoder produces a single syndrome bit. Each array has as input data bits, as when it is encoding. Additionally each array receives as input one and only one parity bit. The receipt of one and only one parity bit by each array allows for the encoder/decoder to be used both for encoding and decoding. This allows a great reduction in hardware required to implement error correction and detection.

EP 0 310 220 A2

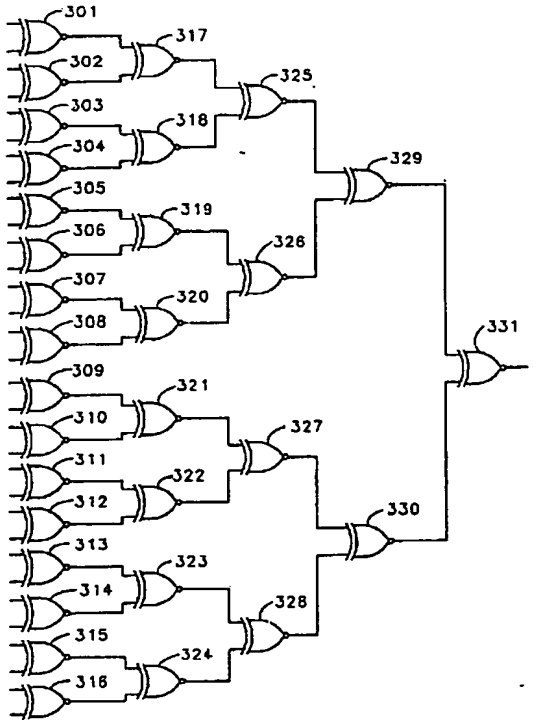

FIG 4

AN APPARATUS USEFUL FOR CORRECTION OF SINGLE BIT ERRORS AND DETECTION OF DOUBLE BIT ERRORS IN THE TRANSMISSION OF DATA

Background

Whenever information is transferred across a bus or other communciation lines, there is the possibility that data may be corrupted. Therefore it is desirable to detect and even correct errors that occur in transmission. In the prior art, Hamming codes are used to correct single bit errors in transmission. This is done by the utilization of parity bits. For a general discussion on the use of parity bits and Hamming codes see Richard E. Blahut, Theory and Practice of Error Control Codes, Edison Wesley, 1983, pp. 11-12, 54-55.

Under the traditional method, for every "n" bits of data, there are required "m" hamming parity bits. The formula for calculating m is the following:

$m = \ln_2(n) + 1$

In addition to single bit error correction, double bit error detection may be achieved by adding a single overall parity bit. Therefore, for one-bit error correction and two-bit error detection m + 1 parity bits need to be used for every n data bits.

For example, if data is being transferred in sixty-four bit words, for every sixty-four (n) bit words of data, seven (m) parity bits must be added in order to correct one-bit errors. m is calculated from the above formula. In order to detect two-bit errors an eighth parity bit must be added.

Table 1, in Figure 1, is used to illustrate the implementation of traditional hamming algorithm. In this implementation, sixty-four bits of information and eight parity bits are used. Each column in Table 1 represents a single line on which is placed a single data bit or a single parity bit. Parity bits are sometimes referred to as code bits, as explained further below.

Row 1 contains letters designating whether each line contains a data bit or a parity bit. If the letter is a "B", the line contains a data bit. If the letter is a "C", the line contains a parity bit. Row 2 and Row 3 number the lines. The parity bits are numbered from 0 to 7. The data bits are numbered from 0 to 63. The parity bits are referred to as C0 - C7. The data bits are referred to as B0 -B63.

As will be explained in greater detail in the Description of the Preferred Embodiment below, lines containing the data bits and the parity bits are coupled into logic circuitry to generate bits P0 - P7, shown in column 13. Bits P0-P7 may be code bits or syndrome bits, as further described below. In this discussion, bits P0-P7 will be assumed to be syndrome bits.

Syndrome bits P0 - P7 are used to detect and correct errors. The are used to detect and correct errors. The information in Rows 4-11 show which data bits and which parity bits are used to calculate each of the syndrome bits P0 - P7. A "1" in the row indicates that a data bit or parity bit is used to calculate the corresponding syndrome bit in Column 13. A "0" in the row indicates that a data bit or parity bit is not used to calculate the corresponding syndrome bit in column 13. For instance, all eight parity bits and all sixty-four data bits are used to generate syndrome bit P0. On the other hand only parity bit C7 and data bits B57, B58, B59, B60, B61, B62 and B63 are used to generate syndrome bit P7.

Column 14 shows the total number of data bits and parity bits to generate each syndrome bit P0 - P7. As can be seen, 72 parity bits and data bits are used to generate syndrome bit P0. 36 parity bits and data bits are used to generate syndrome bit P1. 36 parity bits and data bits are used to generate syndrome bit P2. 36 parity bits and data bits are used to generate syndrome bit P3. 32 parity bits and data bits are used to generate syndrome bit P4. 32 parity bits and data bits are used to generate syndrome bit P5. 32 parity bits and data bits are used to generate syndrome bit P6. 8 parity bits and data bits are used to generate syndrome bit P7.

Row 12 shows how many times each parity bit C0 - C7 and each data it B0 - B63 is used to calculate one of the syndrome bits P0 - P7. As can be seen, the values range from 1 to 7. That is parity bit C0 is used in the calculation of only one syndrome bit --syndrome bit P0--, while data bit B56 is used in the calculation of seven syndrome bits --all but syndrome bit P7--.

The time required to generate each syndrome bit is related to the number of data bits B0-B63 and parity bits C0-C7 used to generate that syndrome bit. This is because, the larger the number of inputs, the more levels of logic must be utilized to generate an output.

Syndrome bit P0 is used as an overall parity bit. In the prior art an overall parity bit has been required for double error detection. 72 data and parity bits are required to generate syndrome bit P0; therefore, in the prior art, in order to perform double error detection, the critical path has been the determination of the overall parity bit, requiring at least one more layer of logic than is required to calculate any of the other syndrome bits P1 - P7. In order to avoid the additional delay, som prior art system, do not have an overall parity bit. That is, in the embodiment shown in Figure 1, syndrom

bit P0 would be omitted and double error detection would not be performed.

Similarly, the number of times each parity bit C0-C7 and each data bit B0-B63 is used to calculate one of the syndrome bits P0-P7 has a notable effect on performance. This is because each input of the circuitry which generates syndrome bits P0 - P7 has a certain amount of capacitance that must be overcome. If a single parity bit or a single data bit is used in the generation of many syndrome bits additional time has to be allowed for the overcoming of the input capacitance.

As may be understood from the preceding discussion, the prior art implementations of hamming algorithms for error correction and detection have not been optimal. The wide variance in the total number of data and parity bits to generate each syndrome bit, and the varying number of times each parity bit and data bit are used to generate a syndrome bit have degraded the performance of these prior art systems. It is an objective of the present invention to present an apparatus which overcome the stated defects of the prior art systems.

Summary of the Invention

In accordance with the preferred embodiment of the present invention an apparatus which corrects single bit errors and detects double bit errors is presented. The present invention includes an encoder/decoder. During encoding the encoder/decoder produces a plurality of parity bits. These parity bits allow the correction of single bit errors in transmission and the detection of double bit errors in transmission. The encoder/decoder comprises a plurality of arrays. Each array consists of logic gates arranged in the form of a binary tree. Each array generates a single parity bit. Further every array receives as input the same number of data bits; however, each array receives a unique combination of data bits. This is an improvement over the prior art. In the prior art, designers did not use equal number of data bits as input to each array. This is especially notable in the generation of overall parity bits in the prior art. In the generation of an overall parity bit all the data bits were used, not just a subset of the data bits.

When the encoder/decoder is used in decoding, the encoder/decoder produces a plurality of syndrome bits. These syndrome bits are used to correct single bit errors in transmission and to detect double bit errors in transmission. When decoding, each array of logic gates within the encoder/decoder produces a single syndrome bit. Each array has as input data bits, as when it is encoding. Additionally each array receives as input one and only one parity bit. The receipt of one and only one parity bit by each array allows for th encoder/decoder to be used for encoding and decoding. This allows a great reduction in hardware required to implement error correction and detection. In the prior art, syndrome bits are generally produced using more than one parity bit. For instance, every parity bit is used in generating the syndrome bit which checks overall parity (P0). As discussed above, when more than one parity bit is used by an array to produce a syndrome bit, different hardware is required for encoding and decoding. Thus the present invention is a significant improvement over the prior art.

Brief Description of the Drawings

Figure 1 shows a hamming matrix which illustrates the implementation of a traditional hamming algorithm.

Figure 2 shows a plurality of memory arrays and two processors coupled to a memory bus.

Figures 3A is a block diagram which shows data flow when data is encoded in accordance with the preferred embodiment of the present invention.

Figures 3B is a block diagram which shows data flow when data is decoded in accordance with the preferred embodiment of the present invention.

Figure 4 shows details of an Encoding/Decoding Block shown in Figures 3A and 3B in accordance with the preferred embodiment of the present invention.

Figures 5A and 5B show additional details of the Encoding/Decoding Block shown in Figures 3A and 3B. Figure 5A shows data flow when data is encoded in accordance with the preferred embodiment of the present invention. Figure 5B shows data flow when data is decoded in accordance with the preferred embodiment of the present invention.

Figure 6 is a block diagram showing data flow in error correction and detection circuitry shown in Figure 3B in accordance with the preferred embodiment of the present invention.

Figures 7A and 7B show alternate embodiments of a correcting decoder shown in Figure 6.

Figure 8 shows details of an error correcting "Exclusive Or" (XOR) array shown in Figure 6.

Figure 9A and 9B show details of an error detecting block shown in Figure 6.

Figure 10 shows a hamming matrix which illustrates the implementation of an apparatus in accordance with the preferred embodiment of the present invention.

Description of the Preferred Embodiment

Figure 2 shows a memory bus 108. Coupled to memory bus 108 is a memory array 111, a memory array 112, a memory array 113, a memory array 114, a memory array 115, a memory array 116, a memory array 117, a memory array 118, a memory array 119, a memory array 120, a memory array 121, a memory array 122, a processor 109 and a processor 110. Processors 109 and 110 access information in memory arrays 111 - 122, through memory bus 108. In order to verify the integrity of data transmitted across data bus 108, the data is encoded before transmission across bus 108, and decoded upon receipt. In the preferred embodiment of the present invention data is organized into sixty-four bit words. After encoding, each sixty-four bit words is accompanied by additional eight code bits when transmitted across bus 108, as further described below.

Figure 3A shows an encoding/decoding block 101. Each processor 109 and 110 and each memory array 111 - 122 has an encoding/decoding block identical to encoding/decoding block 101. In Figure 3A encoding/decoding block 101 is configured to encode data. Sixty-four data bits 102, enter encoding/decoding block 101. Exiting encoding/decoding block 101 are data bits 102, which are unchanged, and eight code bits 104. Data bits 102 and code bits 104 are transmitted together across bus 108. The addition of code bits 104 allows the correction of a single error in transmission of data bits 102 and for detection of two errors in transmission of data bits 102 and code bits 104.

Figure 3B shows encoding/decoding block 101 configured for decoding data. Upon reception of a transmission across bus 108 encoding/decoding block 101 receives sixty-four data bits 102 and eight code bits 104. Data bits 102 and code bits 104 may have been corrupted by transmission through bus 108. Exiting encoding/decoding block 101 are sixty-four data bits 102, which encoding/decoding block 101 does not alter, and eight syndrome its 204. Syndrome bits 204 are used by error correction and detection circuitry 203. Error correction and detection circuitry 203 receives syndrome bits 204 and uncorrected data bits 102. Error correction and detection circuitry 203 produces sixty-four corrected data bits 202, a multiple bit error signal 606 and an error signal 607.

If there is no error in transmission, error signal 607 will be at logic 1. If there is an error in transmission, error signal 607 will be at logic 0. If there is no error in transmission, or two errors in transmission, multiple bit error signal 606 will be at logic 1. If ther is a single rror in transmission, error signal 607 will be at logic 0. If error signal 607 is at logic 0 and multiple bit error signal 606 is at logic 1, it is assumed that there were two errors in transmission.

If error correction and detection circuitry 203 detects no error in transmission, data bits 202 are identical to data bits 102. If error correction and detection circuitry 203 detects a single error in transmission, data bits 202 differ from data bits 102 in the bit in which the error occurred. The uncorrupted bits of data bits 102 will be identical with the corresponding bits of data bits 202. If error correction and detection circuitry 203 detects two errors in transmission, data bits 202 are thrown away and retransmission is required.

Figure 4 shows an array of Exclusive-Nor gates 301-331 coupled together in a binary tree structure. The binary tree structure of Exclusive-Nor gates is referred to herein as an XNOR array. The XNOR array shown in Figure 4 has thirty-two inputs and one output.

Figure 5A shows details of Encoding/Decoding Block 101 shown in Figures 3A and 3B. Figure 5A shows data flow when data is encoded in preparation for transmission across bus 108. A subset of data bits 102 serves as input to an XNOR array 400, an XNOR array 401, an XNOR array 402, an XNOR array 403, an XNOR array 404, an XNOR array 405, an XNOR array 406 and an XNOR array 407. Each of XNOR arrays 400 - 407 receives thirty of sixty-four data bits from data bits 102. However, no two XNOR array from XNOR arrays 400-407 receive exactly the same thirty data bits. The means for selecting which data bits a particular XNOR array receives is discussed later in the explanation of Table 2 in Figure 11.

XNOR arrays 400 - 407 are identical to the XNOR array shown in Figure 4. When XNOR arrays 400 - 407 are configured as shown in Figure 5A, only thirty of thirty-two inputs of each of XNOR arrays 400-407 are utilized. The other two inputs are set at logic 0.

Each of XNOR arrays 400-407 produces a single output. These outputs from XNOR arrays 400-407 are the eight code bits 104.

Figure 5B similarly shows details of Encoding/Decoding Block 101 shown in Figures 3A and 3B. Figure 5B shows data flow when data is decoded after a transmission is received from bus 108. A subset of data bits 102 serves as input to XNOR array 400, XNOR array 401, XNOR array 402, XNOR array 403, XNOR array 404, XNOR array 405, XNOR array 406, and XNOR array 407. Each of XNOR arrays 400 - 407 receives thirty of sixty-four data bits from data bits 102. However, no two XNOR array from XNOR arrays 400-407 receive exactly the same thirty data bits. Additionally,

each of XNOR arrays 400 - 407 receives a code bit from code bits 104. The means for selecting which data bits from data bits 102, and which code bits from code bits 104 a particular XNOR array receives is discussed later in the explanation of Table 2 in Figure 11.

When XNOR arrays 400 - 407 are configured as shown in Figure 5A, only thirty-one of thirty-two inputs of each of XNOR arrays 400-407 are utilized. The final input is set at logic 0. In the preferred embodiment of the present invention the final input of each of XNOR arrays 400 - 407 is not utilized in normal operation. These may be used for diagnostic purposes to verify correct function of the system circuitry.

Each of XNOR arrays 400-407 produces a single output. These outputs from XNOR arrays 400-407 are the eight syndrome bits 204.

Figure 6 is a more detailed block diagram of error correction and detection circuitry 203 shown in in Figure 3B. Syndrome bits 204 enter an error correcting decoder 601 and an error detection block 602. Additionally the complement of each of syndrome bits 204 is produced by placing syndrome bits 204 through inverter 603 to produce complemented syndrome bits 604. Complemented syndrome bits 604 also serve as input to error correcting decoder 601. Error correcting decoder 601 produces sixty-four flipper control bits 608.

Each of flipper control bits 608 corresponds to a data bit of data bits 102. When there is a single error in transmission across bus 108, flipper control bits will indicate in which bit there is an error. When there is no detected error in transmission each of flipper control bits 608 will be at logic 0. When there is a single bit error, for the bit of data bits 102 in which the single error occurs, the corresponding flipper control bit from flipper control bits 608 will be at logic 1.

An error correction array of "Exclusive-Or" gates (XOR array) 605 receives flipper control bits 608 and data bits 102. Error correction XOR array 605 combines data bits 105 and flipper control bits 608 to produce corrected data bits 202.

Error detection block 602 receives syndrome bits 204 and produces multiple bit error signal 606 and error signal 607.

Figure 7A shows details of the implementation of error correcting decoder 601. Error correcting decoder 601 comprises sixty-four gates of which one is shown in Figure 7. A gate 701, complements its eight inputs and performs a logic "AND" operation. Gate 701 produces a single flipper control bit 608a. Flipper control bit 608a is one of sixty-four flipper control bits 608. Each of eight inputs of gate 701 is coupled to one of syndrome bits 204 or complem nted syndrome bits 604, as shown. A gate similar to gate 701 is used to produce each flipper control bit of flipper control bits 608. Each gate similarly generates a flipper control bit of flipper control bits 608 using inputs from syndrome bits 204 and complemented syndrome bits 604. Which combination of syndrome bits 204 and complemented syndrome its 604 are utilized to produce each of flipper control bits 608 will be addressed in the discussion of Table 2 shown in Figure 11.

Figure 7B shows an alternate embodiment. In Figure 7B a five input gate 702 produces a single flipper control bit 608b. Each error may be uniquely identified from five bits from syndrome bits 204 and/or complemented syndrome bits 604. Therefore error correcting decoder 601 may be implemented using five input gates such as gate 702.

Figure 8 shows details of the implementation of error correction XOR array 605. Error correction XOR array 605 comprises sixty-four XOR gates. A single XOR gate 804 is shown in Figure 8 as an example. XOR gate 804 receives a data bit 102a from data bits 102 and a corresponding flipper control bit 608a from flipper control bits 608. XOR gate 804 produces a corresponding corrected data bit 202a of corrected data bits 202. In this way all sixty four corrected data bits 202 are generated. As has been discussed above, when there are no errors in transmission, each of flipper control bits 608 will be at logic 0, and corrected data bits 202 will be identical to data bits 102. If there is an error in a single bit, the corresponding flipper control bit will be at logic 1, and the data bit in error will be corrected. In the shown embodiment XOR gates have been used. However, with only slight changes XNOR gates may be used instead, and may be preferred in certain cases.

Figure 9A and Figure 9B show details of an error detecting block shown in Figure 6. In Figure 9A an array of XNOR gates comprising an XNOR gate 901, an XNOR gate 902, an XNOR gate 903, an XNOR gate 904, an XNOR gate 905, an XNOR gate 906 and an XNOR gate 907 are coupled together to form a binary tree, as shown. Syndrome bits 202 are input to the array of XNOR gates 901 - 907. The array of XNOR gates 901 - 907 produces multiple error signal 606. As can be seen, if an odd number of syndrome bits 202 are at logic 1, multiple error signal 606 will be at logic 0. Otherwise multiple error signal 606 will be at logic 1.

If there are no errors in transmission across bus 108, syndrome bits 202 will all be at logic 0. In this case multiple error signal 606 will be at logic 1. If there is an error in a single bit transmitted across bus 108, an odd number of syndrome bits 202 will be at logic 1. In this case multiple error signal 606 will be at logic 0. If there are errors in two bits transmitted across bus 108, an even numb r of

syndrome bits 202 will be at logic 1. In this case multiple error signal 606 will be at logic 1.

In Figure 9B is shown an eight-input gate 910. Gate 910 complements the value of each of its inputs and then performs a logic AND operation. The operation performed by gate 910 is equivalent to a logic "NOR" operation. Gate 910 produces error signal 607. When all the inputs to gate 910 are at logic 0, error signal 607 wil be at logic 1. Otherwise error signal 607 will be at logic 0.

Syndrome bits 202 serve as input to gate 910. If there is no error in transmission across bus 108, syndrome bits 202 will all at logic 0. In this case error signal 607 will be at logic 1. If one or more errors are detected in a transmission across bus 108, not all of syndrome bits 202 will be at logic 0. In this case error signal 607 will be at logic 0. When error signal 607 is at logic 0, this indicates an error in transmission has been detected.

Table 2 in Figure 10 gives implementation detail concerning the present invention, particularly regarding implementation of XNOR arrays 400 - 407 within encoding/decoding block 101 and regarding implementation of error correcting decoder 601.

In Figure 5A and Figure 5B XNOR arrays 400 - 407 each receive input from thirty of the sixty-four data bits 102. The following is an explanation of how, for the preferred embodiment, to determine which data bits are input into which XNOR array.

With the exception of columns 1013 and 1014, each column in Table 2 represents a single data bit from data bits 102, or a single parity bit.

Row 1001, at the top of each column, contains letters designating whether each line contains a data bit from data bits 102 or a parity bit. If the letter is "B", the line contains a data bit from data bits 102. If the letter is a "C", the line contains a parity bit. Row 1002 and Row 1003 identify particular data bits from data bits 102, and identify particular parity bits. In row 1003, the parity bits are numbered from 0 to 7. The parity bits therefore may be referred to using the designations C0 - C7. In row 1002 and 1003, data bits 102 are numbered from 0 to 63. Data bits 102 therefore may be referred to using the designations B0 - B63.

In column 1013 are shown bits P0, P1, P2, P3, P4, P5, P6 and P7. Bits P0 - P7 represent the output of XNOR arrays 400-407 shown in Figures 5A and 5B. XNOR array 400 produces bit P0, XNOR array 401 produces bit P1, XNOR array 402 produces bit P2, XNOR array 403 produces bit P3, XNOR array 404 produces bit P4, XNOR array 405 produces bit P5, XNOR array 406 produces bit P6 and XNOR array 407 produces bit P7.

Rows 1020, 1021, 1022, 1023, 1024, 1025, 1026 and 1027 indicate which data bits B0 - B63 of data bits 102 are input to which of XNOR arrays 400 - 407. A "1" in row 1020 indicates a bit is an input to XNOR array 400. A "1" in row 1021 indicat s a bit is an input to XNOR array 401. A "1" in row 1022 indicates a bit is an input to XNOR array 402. A "1" in row 1023 indicates a bit is an input to XNOR array 403. A "1" in row 1024 indicates a bit is an input to XNOR array 404. A "1" in row 1025 indicates a bit is an input to XNOR array 405. A "1" in row 1026 indicates a bit is an input to XNOR array 406. A "1" in row 1027 indicates a bit is an input to XNOR array 407. A "0" indicates a data bit is not an input to the XNOR array in the corresponding row. For example, from Table 2, data bit B63 is an input to XNOR array 400, XNOR array 403 and XNOR array 404, but is not an input to XNOR array 401, XNOR array 402, XNOR array 405, XNOR array 406 nor XNOR array 407.

Each XNOR array of XNOR arrays 400 - 407 receives thirty data bits from data bits 102. In addition each XNOR array receives a single parity bit from parity bits C0 - C7. The thirty-second input to each of XNOR array 400-407 is at logic 0 except when diagnostics are performed on the system.

When encoder/decoder block 101 is configured as in Figure 3A and Figure 5A, bits P0 - P7 represent code bits 104. In the preferred embodiment, in this configuration the parity bits are all 0. Therefore, in the configuration shown in Figures 3A and 5A, the two non-data inputs in each of XNOR arrays 400-407 are at logic 0.

When encoder/decoder block 101 is configured as in Figure 3B and Figure 5B, bits P0 - P7 represent syndrome bits 204. The parity bits C0 - C7 are code bits 104. Code bits are placed in encoder/decoder block 101 in the same in order in which they are generated when encoder/decoder block 101 is configured as in Figures 3A and 5A.

That is, when encoder/decoder block 101 is configured as in Figures 3A and 5A, code bits 104 are generated by XNOR arrays 400 - 407 and referred to as outputs P0 - P7. When encoder/decoder block 101 is configured as in Figures 3B and 5B, code bits 104 are used as input to encoder/decoder block 101 and referred to as parity bits C0 - C7. If a code bit from code bits 104 was labelled P0 when it was generated when encoder/decoder block 101 is configured as in Figure 3A, then the code bit from code bits 104 is labelled C0 when it serves as input to encoder/decoder block 101 when configured as in Figure 3B. If a code bit from code bits 104 was labelled P1 when it was generated when encoder/decoder block 101 is configured as in Figure 3A, then the code bit from code bits 104 is labelled C1 when it serves as input to encoder/decoder block 101 when configured as in Figure 3B. And so on.

Table 2 may also be used to implement error correcting decoder 601. Tabl 2 shows which combination of syndrome bits 204 and complemented syndrome bits 604 are utilized to produce each of flipper control bits 608. This may be illustrated by observation of the inputs to gate 701 and the contents of the column under data bit B63. Gate 701 produces the flipper bit for data bit B63. For the "1"s in the column under data bit B63, the corresponding complemented syndrome bit 604 is an input to gate 701. For the "0"s in the column under data bit B63, the corresponding syndrome bit 204 is an input to gate 701.

Similarly, for the five-input system in Figure 7B, gate 702 produces the flipper bit for data bit B62. The flipper bit for data bit B62 is uniquely identified by the "1"s in the column under data bit B62. Therefore, in order to determine the flipper bit for data bit B62, for ever "1" in the column under data bit B62, the corresponding complemented syndrome bit of syndrome bits 604 is an input to gate 702.

Column 1014 shows the total number data bits 102 and parity bits P0 -P7 used as input into each XNOR arrays of XNOR arrays 400 - 407. Each of XNOR arrays 400 - 407 receives a total of 31 data and parity bits. This may be compared to the contents of row 14 in Table 1 of Figure 1. As can be seen the present invention gives a notable improvement. The maximum inputs for the present invention is less than have the maximum input of the prior art. The present invention, therefore can be implemented with two fewer levels of logic.

Row 1012 shows how many times each parity bit C0 - C7 and each data bit from data bits 102 is used as an input into an XNOR array from XNOR arrays 400 - 407. Each data bit from data bits 102 is used either three of five times. Each code bit is used only once. This is an improvement over the information in Row 12 in Table 1 of Figure 1, where a single data bit is used up to seven times. This is because each input of XNOR array 400 - 407 has a certain amount of capacitance that must be overcome. The more XNOR arrays from XNOR arrays 401 - 407 for which a single code bit or a single parity bit is used as input the greater total amount of input capacitance needs to be overcome. In the prior art, the greatest number was seven. The present invention, therefore is an improvement.

In order to avoid "glitches" in the system, slowing performance, "dummy loads" may be added to balance the input. Thus if a data bit is used as input to three of XNOR arrays 401 - 407, two "dummy loads" may be added to simulate the load as if the data bit was used by five of XNOR arrays 401 - 407. Similarly, each parity bit is used only once, so four dummy loads could be added to each. Of course no dummy loads should be added to data bits which already serve as input to five XNOR arrays from XNOR arrays 401 - 407.

The information in Table 2 was generated using the following constraints. In order to facilitat the production of tables similar to Table 2, it is helpful to utilize a computer in the selection of table entries in conformance with the following constraints.

Constraint 1: Each of XNOR arrays 400 - 407 receives a single parity bit.

Constraint 2: Each data bit from data bits 102 is used as input for a unique combination of either three or five XNOR arrays from XNOR arrays 400 -407.

Constraint 3: Each XNOR array from XNOR arrays 400 -407 receives a unique combination of 30 data bits from data bits 102 as input.

Constraint 4: Data bits 102 are divided into pairs, e.g., B00 and B01 are a pair, B02 and B03 are a pair, etc. Only one data bit from each pair enters a particular XNOR array from XNOR arrays 400 - 407.

Constraint 1 assures that encoder/decoder block 101 can be used both for encoding and decoding. Constraint 2 assures that overall parity is preserved. Constraint 3 in combination with Constraint 2 assures that each syndrome bit has a signature. Therefore, if there is a single bit error it is possible to correct the error. Constraint 4 makes routing easier. This helps in the actual implementation allowing this Encoding/Decoding block 101 to be hardwired programmable like a read only memory (ROM).

Claims

1. An apparatus for assuring the integrity of a first plurality of data bits when the first plurality of data bits has been transmitted accompanied with a plurality of parity bits, the apparatus comprising:
syndrome generation means for generating a plurality of syndrome bits, the syndrome generation means divided into a plurality of arrays each array producing a single syndrome bit of the plurality of syndrome bits, each array receiving a subset of the first plurality of data bits, and each array receiving a single parity bit so that only one parity bit is used in the generation of each syndrome bit; and,
error correction and detection means, coupled to the syndrome generation means, for receiving the plurality of data bits and the plurality of syndrome bits, for correcting an error in the first plurality of data bits when there is an error in the transmission of a single bit from the first plurality of data bits

and for detecting when there has been an error in the transmission of two bits from the first plurality of data bits.

2. An apparatus as in Claim 1, wherein each array of the plurality of arrays receives the same number of data bits from the plurality of data bits, but receives a different combination of data bits from the plurality of data bits.

3. An apparatus for insuring the integrity of transmission of a first plurality of data bits, the apparatus comprising:

encoding means for receiving the plurality of data bits and producing a plurality of parity bits, the encoding means divided into a plurality of arrays each array producing a single parity bit of the plurality of parity bits, and wherein each array of the plurality of arrays receives the same number of data bits from the plurality of data bits, but receives a different combination of data bits from the plurality of data bits.

transmission means, coupled to the encoding means, for transmitting the plurality of data bits and the plurality of parity bits;

syndrome generation means, coupled to the transmission means, for generating a plurality of syndrome bits from the plurality of data bits and from the plurality of parity bits: and,

error correction means, coupled to the syndrome generation means, for receiving the plurality of data bits and the plurality of syndrome bits and for correcting an error in the first plurality of data bits when there is an error in the transmission of a single bit from the first plurality of data bits.

4. An apparatus as in Claim 3, wherein the syndrome generation means is identical to the encoding means.

5. An apparatus as in Claim 4

wherein the error correction means includes an error detection means for receiving the plurality of syndrome bits and for detecting when there is an error in the transmission of two bits of the first plurality of data bits,

wherein each syndrome bit has a corresponding parity bit from the plurality of parity bits, and

wherein the syndrome generation means uses the corresponding parity bit in the generation of a syndrome bit but does not use any other parity bit for the generation of that syndrome bit.

6. An encoder for generating m + 1 parity bits, from n data bits, the m + 1 parity bits allowing single error correction and double error detection in the n data bits after transmission, the relationship of n to m being given by the formula

$m = \ln_2(n) + 1$

the encoder comprising:

a plurality of m + 1 arrays, ach array in the plurality of m + 1 producing a single parity bit of the m + 1 parity bits, each array in the plurality of m + 1 arrays being identical in structure to every other array in the plurality of m + 1 arrays, each array comprising a plurality of logic gates arranged in a binary tree structure and each array receiving as input a unique combination of p data bits from the plurality of data bits, wherein p is a positive integer and wherein p is the same for each array in the plurality of m + 1 arrays.

7. An encoder as in Claim 6 wherein m equals eight, n equals sixty-four and p equals thirty.

8. An encoder as in Claim 6 wherein each data bit from the n data bits is used as input to at least three arrays from the plurality of m + 1 arrays, is used as input to at most five arrays from the plurality of m + 1 arrays, is not used as input in exactly four arrays from the plurality of m + 1 arrays, and is used as input to a unique combination of arrays from the m + 1 arrays.

9. A decoder for generating m + 1 syndrome bits from m + 1 parity bits and n data bits, the m + 1 syndrome bits sufficient to correct single errors and detect double errors in the transmission of the n data bits, the relationship of n to m being given by the formula

$m = \ln_2(n) + 1$

the decoder comprising:

a plurality of m + 1 arrays, each array in the plurality of m + 1 arrays producing a single syndrome bit from the m + 1 syndrome bits, each array in the plurality of m + 1 arrays being identical in structure to every other array in the plurality of m + 1 arrays, each array comprising a plurality of logic gates arranged in a binary tree structure, each array receiving as input a unique combination of data bits from the plurality of data bits, and each array receiving as additional input one and only one parity bit from the m + 1 parity bits.

10. A decoder as in Claim 9 wherein each array from the m + 1 arrays receives p data bits from the plurality of data bits, wherein p is a positive integer and wherein p is the same for each array in the plurality of m + 1 arrays.

11. A decoder as in Claim 10 wherein m equals eight, n equals sixty-four and p equals thirty.

12. A decoder as in Claim 11 wherein each data bit from the n data bits is used as input to at least three arrays from the plurality of m + 1 arrays, is used as input to at most five arrays from the plurality of m + 1 arrays, is not used as input in exactly four arrays from the plurality of m + 1 arrays, and is used as input to a unique combination of arrays from the m + 1 arrays.

13. An apparatus for insuring the integrity of transmission of data, the apparatus comprising:

encoding/decoding means for receiving the a plurality of data bits and a first plurality of parity bits, and for producing a second plurality of parity bits, the encoding/d coding means divided into a plural-

ity of arrays each array receiving one and only one parity bit from the first plurality of parity bits, each array receiving data bits from the plurality of data bits, and each array producing one and only on of the parity bits from the second plurality of parity bits; wherein when the encoding/decoding means is encoding, the first plurality of parity bits are all at logic 0 and the second plurality of parity bits are code bits, and wherein when the encoding/decoding means is decoding, the first plurality of parity bits are code bits and the second plurality of parity bits are syndrome bits;

transmission means, coupled to the encoding/decoding means, for transmitting the plurality of data bits and the second plurality of parity bits when encoding/decoding means is encoding and the second plurality of parity bits are code bits;

error correction means, coupled to the encoding/decoding means, for, when the encoding/decoding means is decoding, receiving the plurality of data bits and the plurality of syndrome bits, and for correcting an error in the first plurality of data bits when there is an error in the transmission of a single bit from the first plurality of data bits.

14. An apparatus as in Claim 13, wherein each array of the plurality of arrays receives the same number of data bits from the plurality of data bits, but receives a different combination of data bits from the plurality of data bits.

TABLE 1

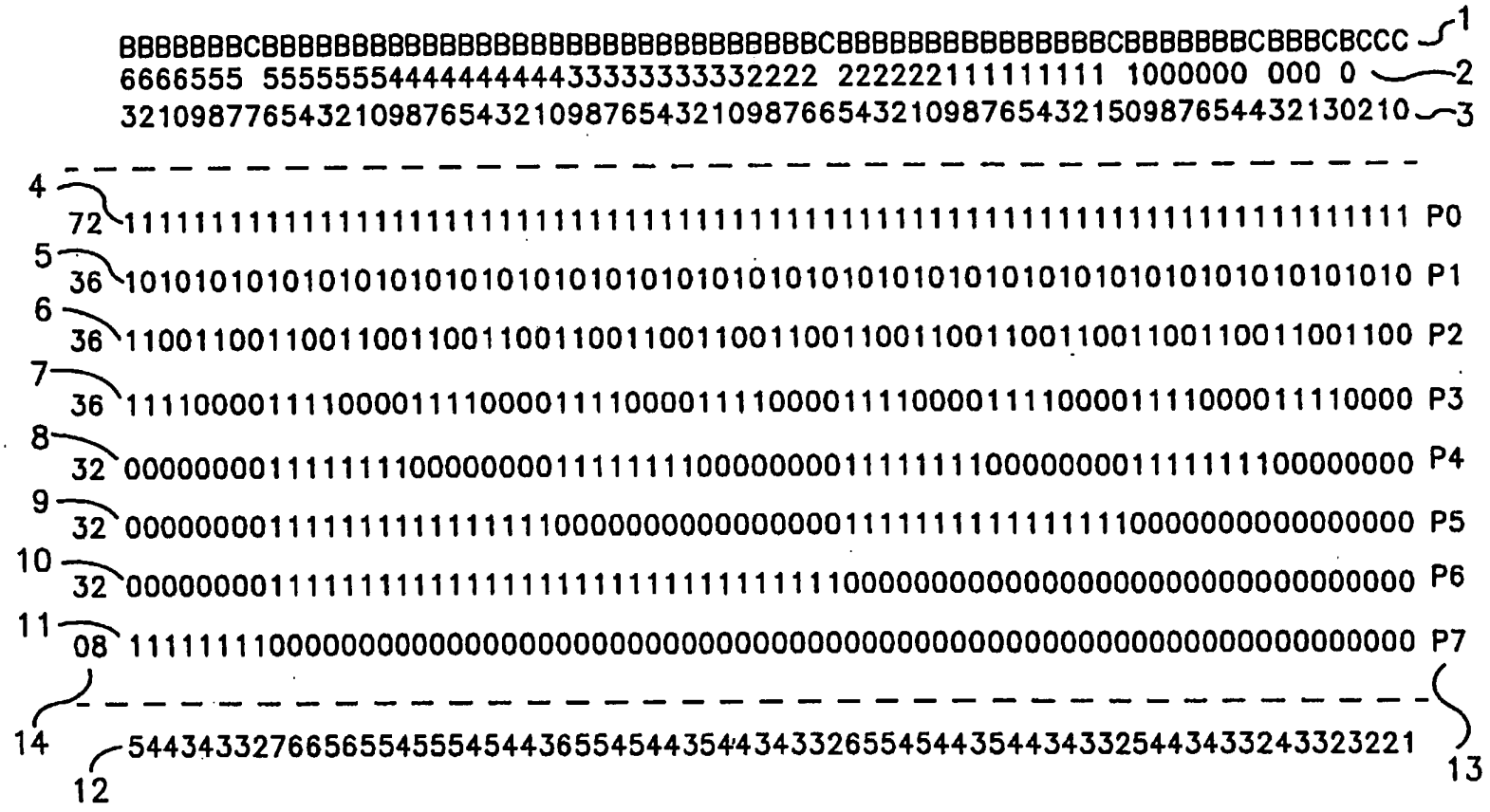

FIG 1 (PRIOR ART)

EP 0 310 220 A2

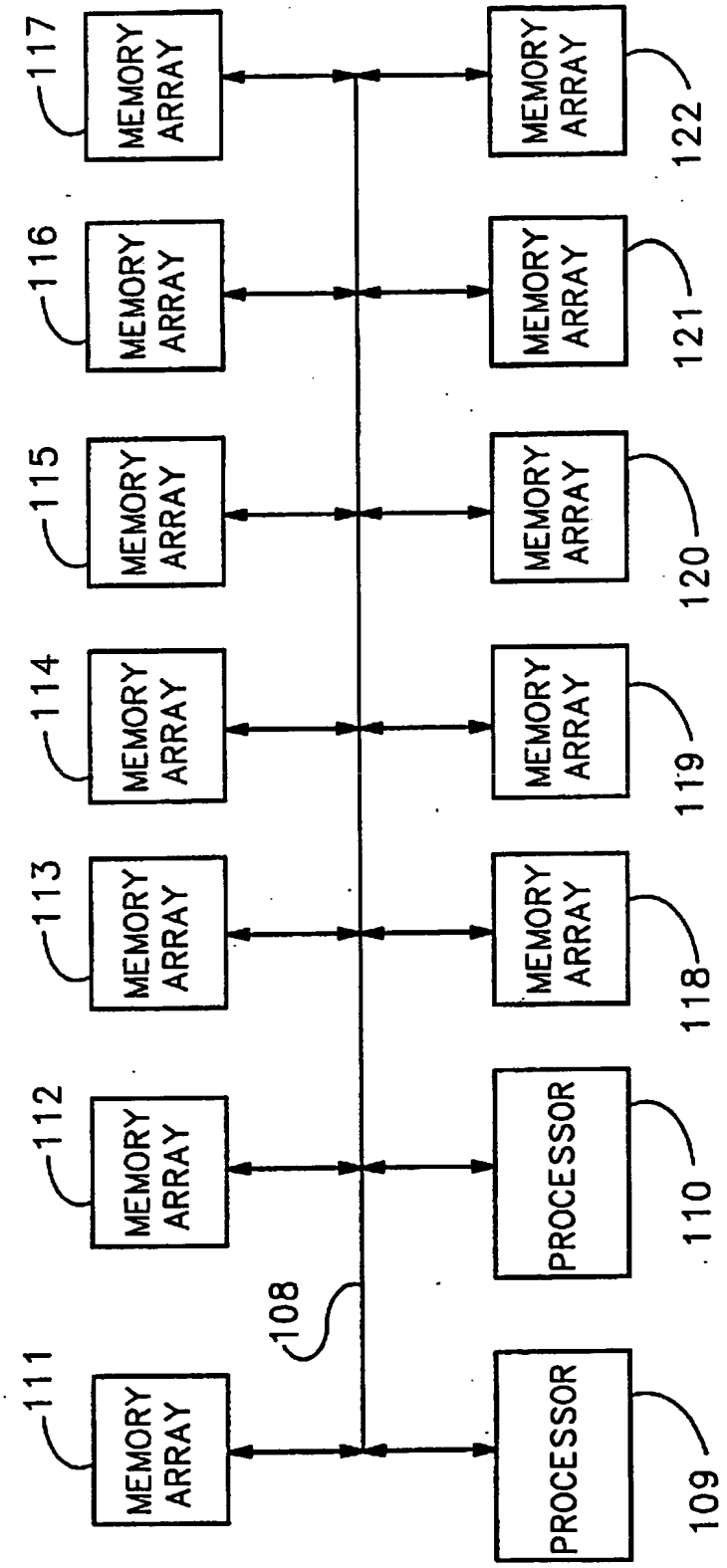
MEMORY ARRAY
PROCESSOR
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122

FIG 2

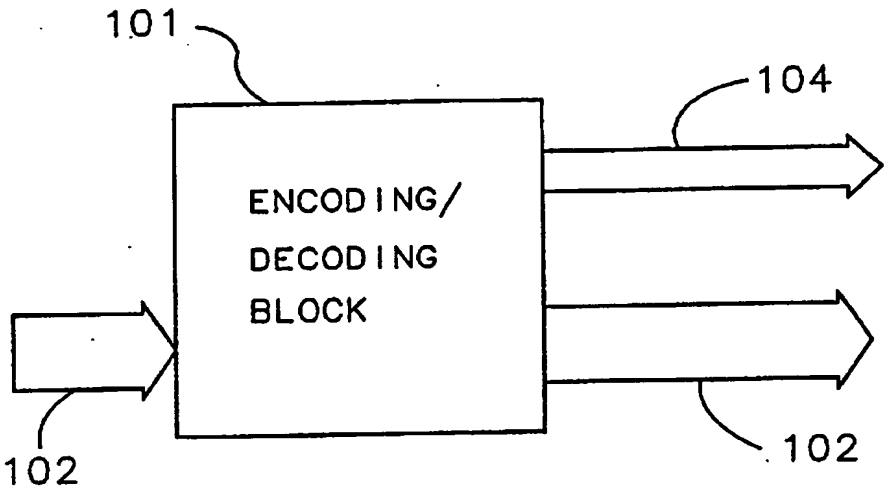
101
104
ENCODING/
DECODING
BLOCK
102
102

FIG 3A

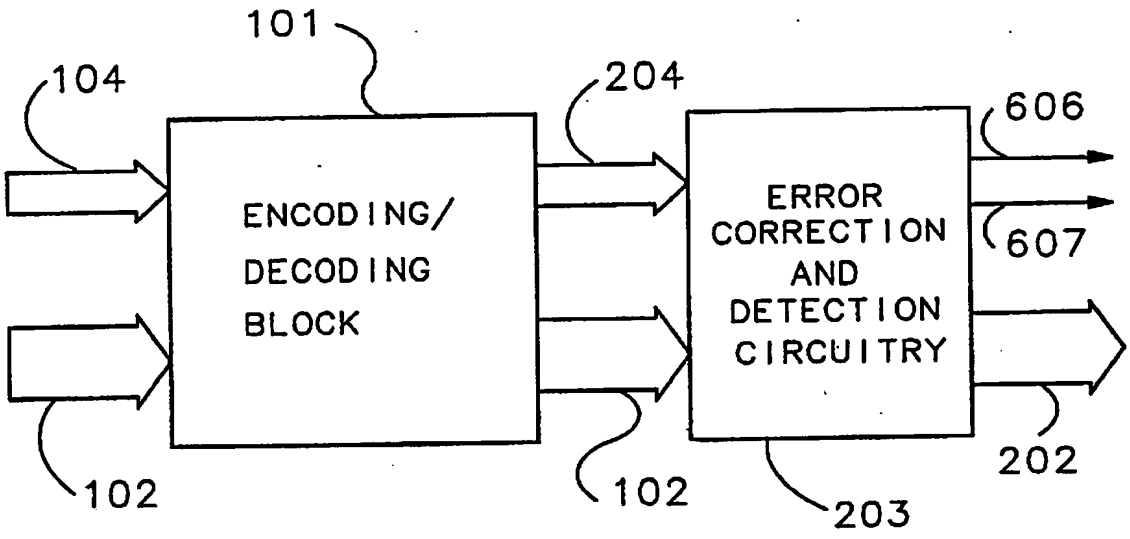
101
104
204
606
ENCODING/
DECODING
BLOCK
ERROR
CORRECTION
AND
DETECTION
CIRCUITRY
607
202
102
102
203

FIG 3B

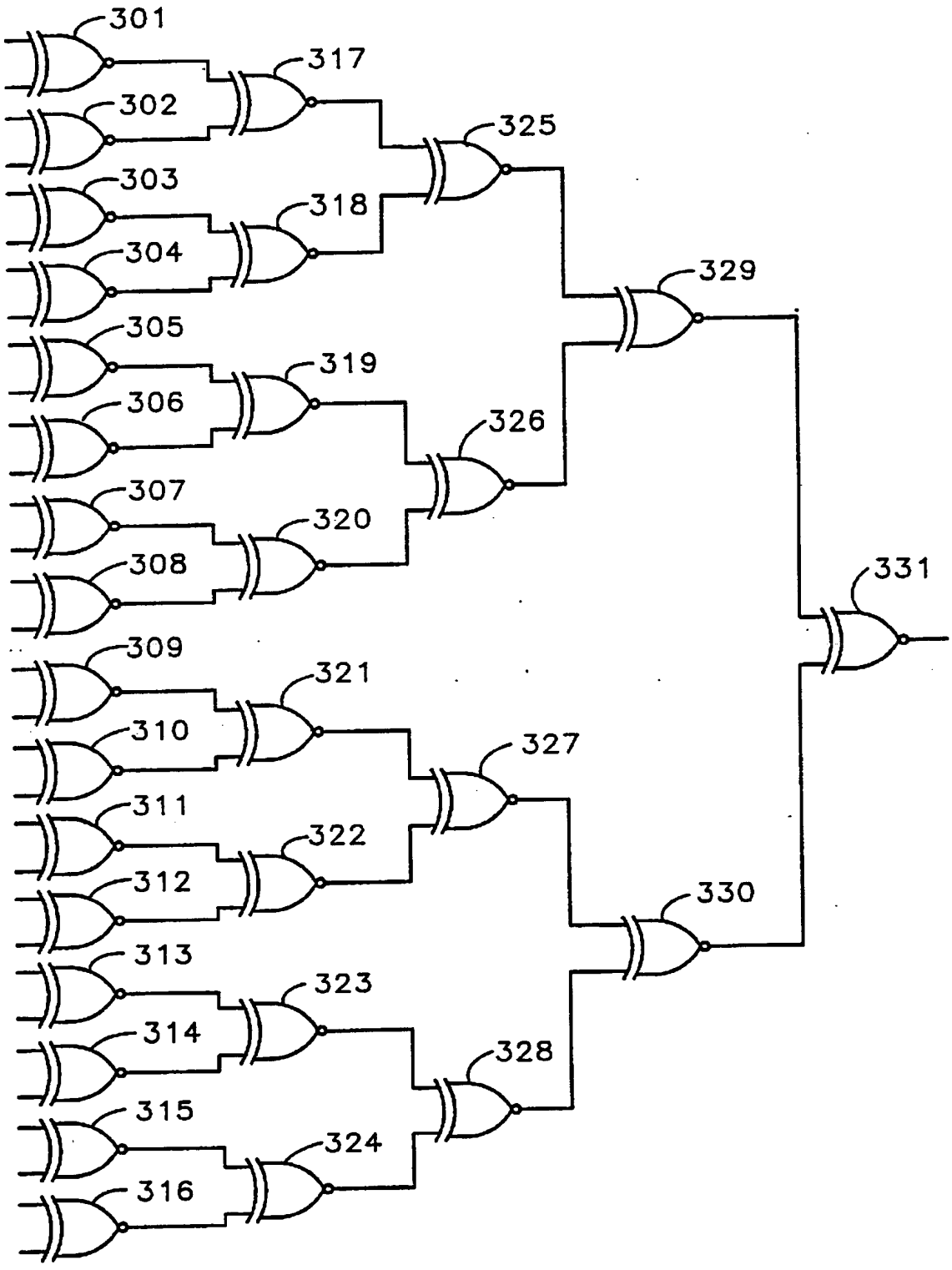
301
317
302
325
303
318
304
329
305
319
306
326
307
320
308
331
309
321
310
327
311
322
312
330
313
323
314
328
315
324
316

FIG 4

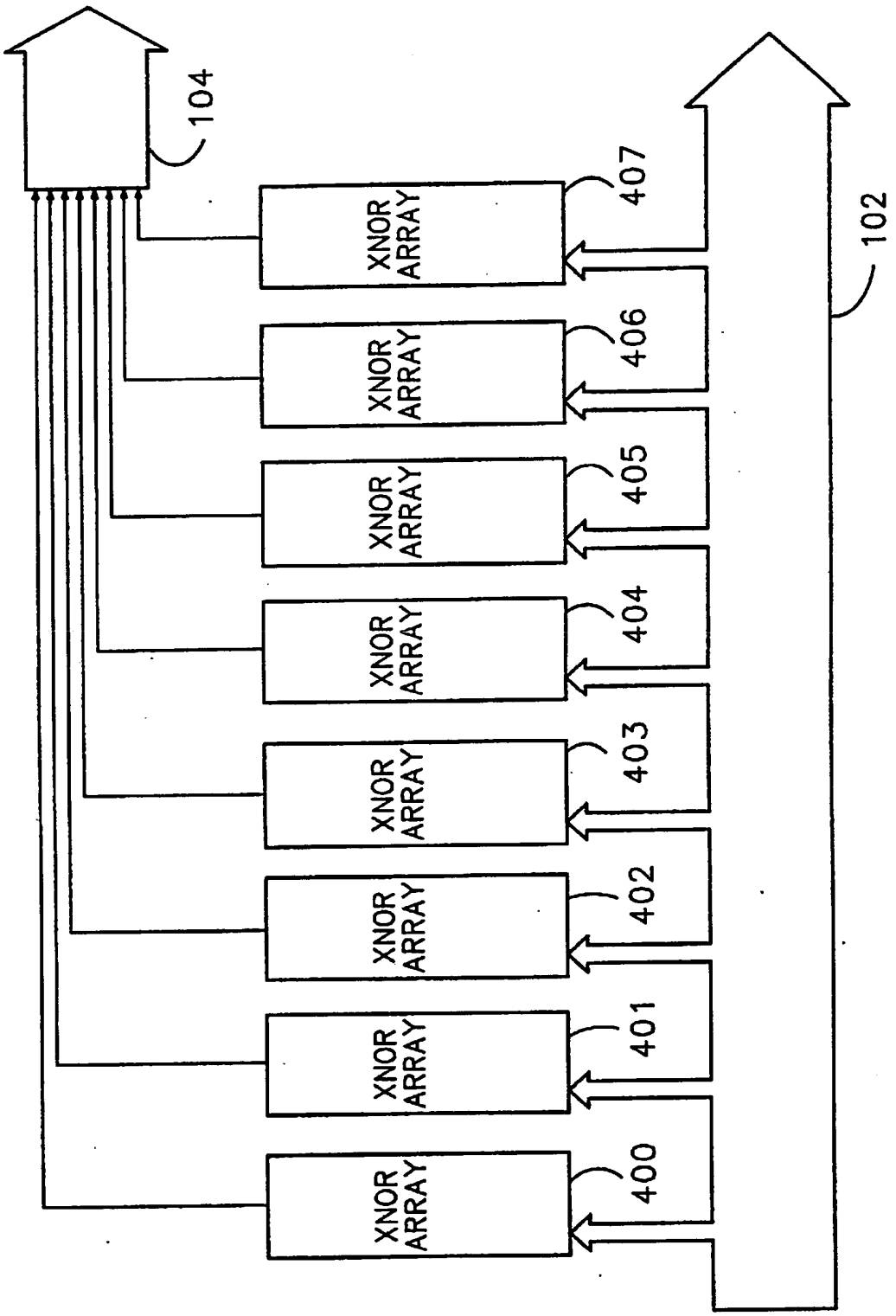

FIG 5A

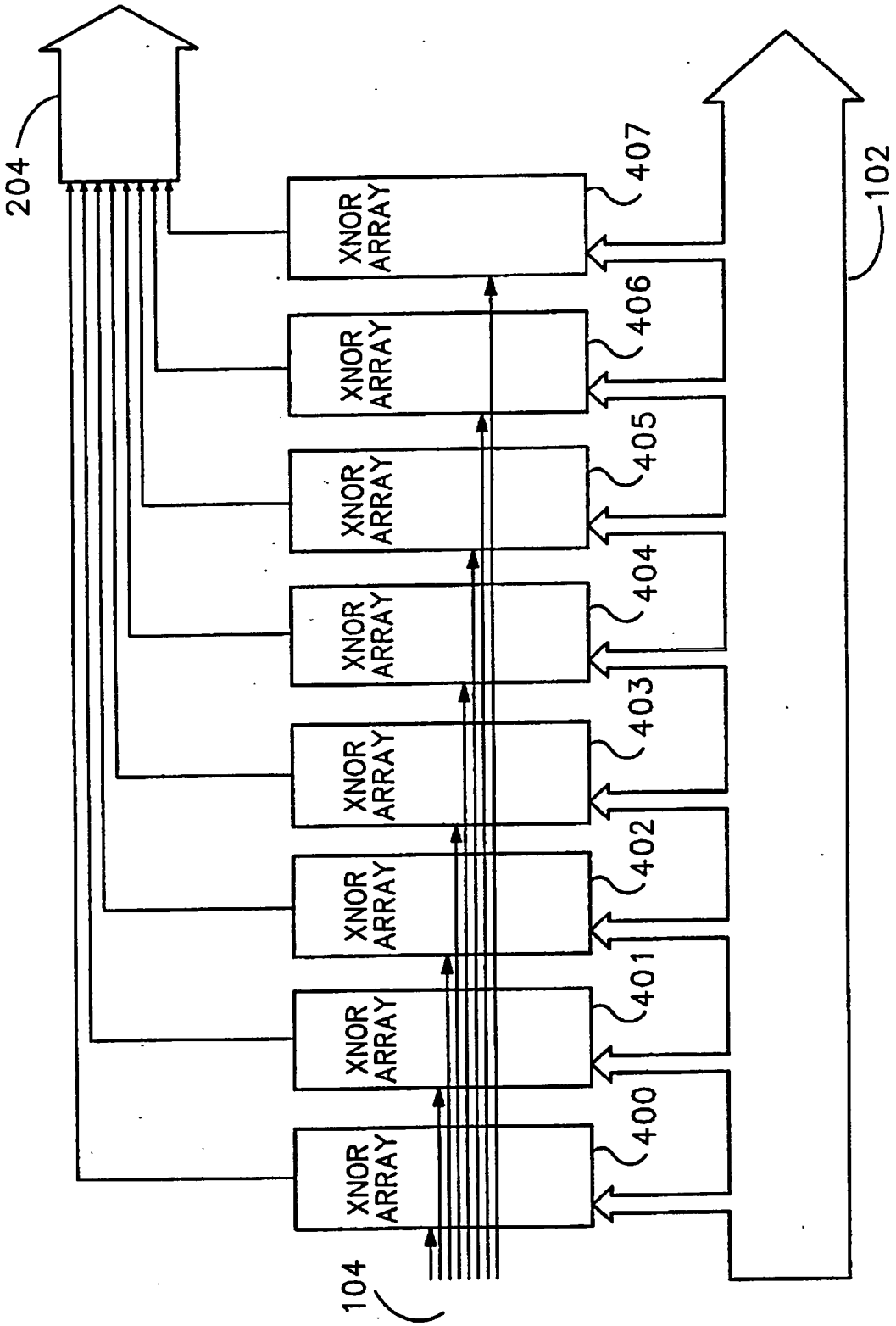

FIG 5B

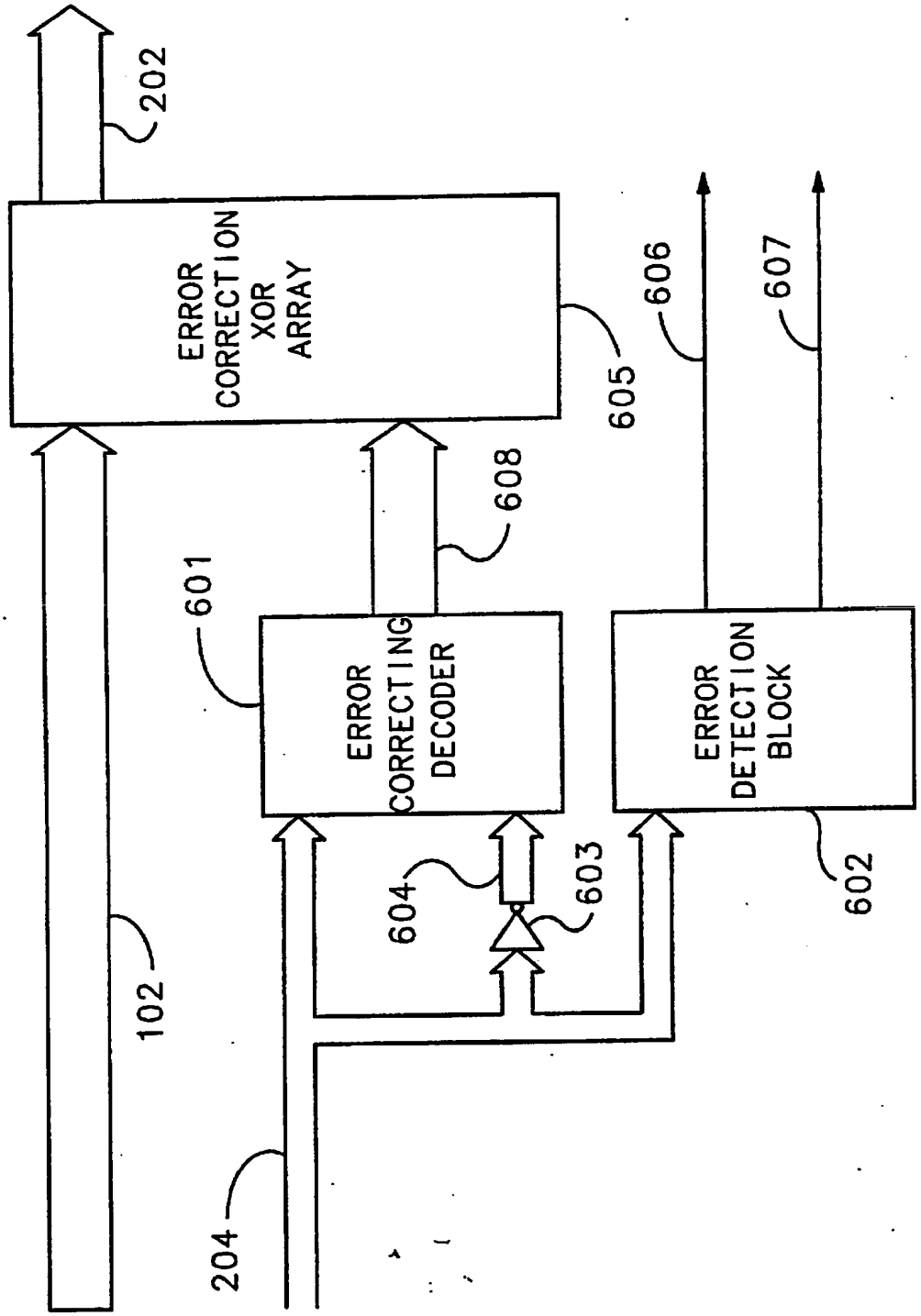
202
ERROR CORRECTION XOR ARRAY
605
606
607
608
601
ERROR CORRECTING DECODER
ERROR DETECTION BLOCK
602
604
603
102
204

FIG 6

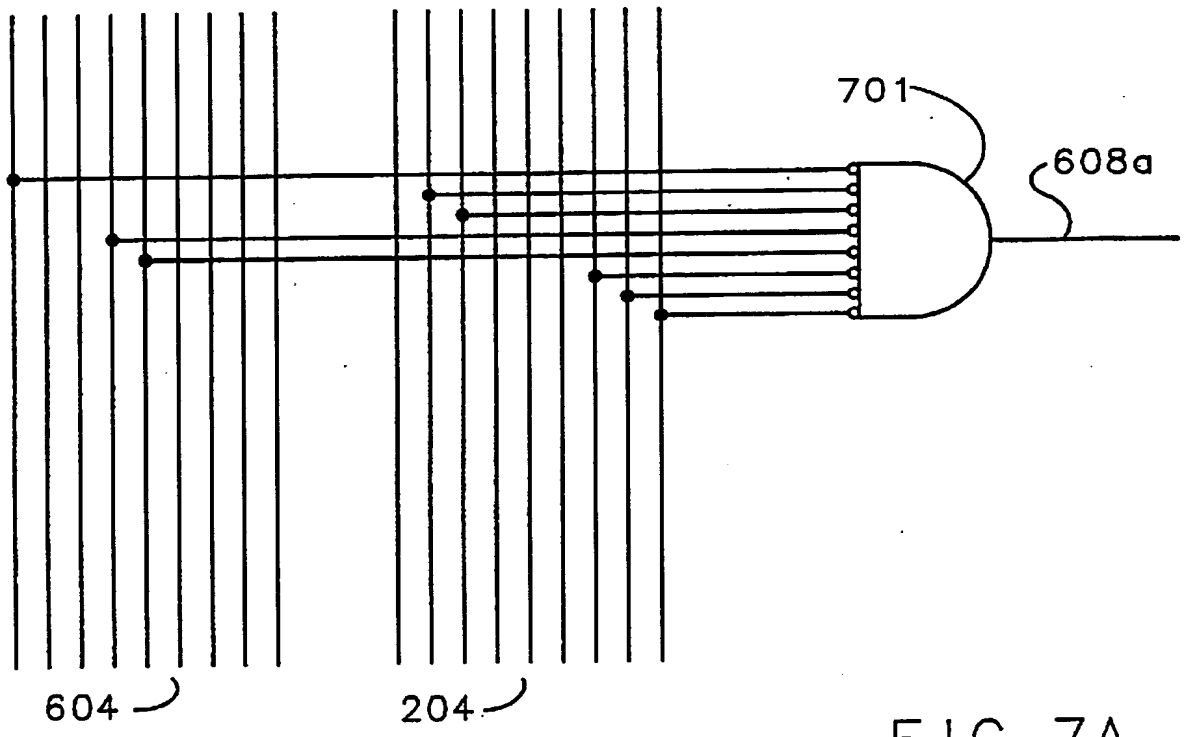

FIG 7A

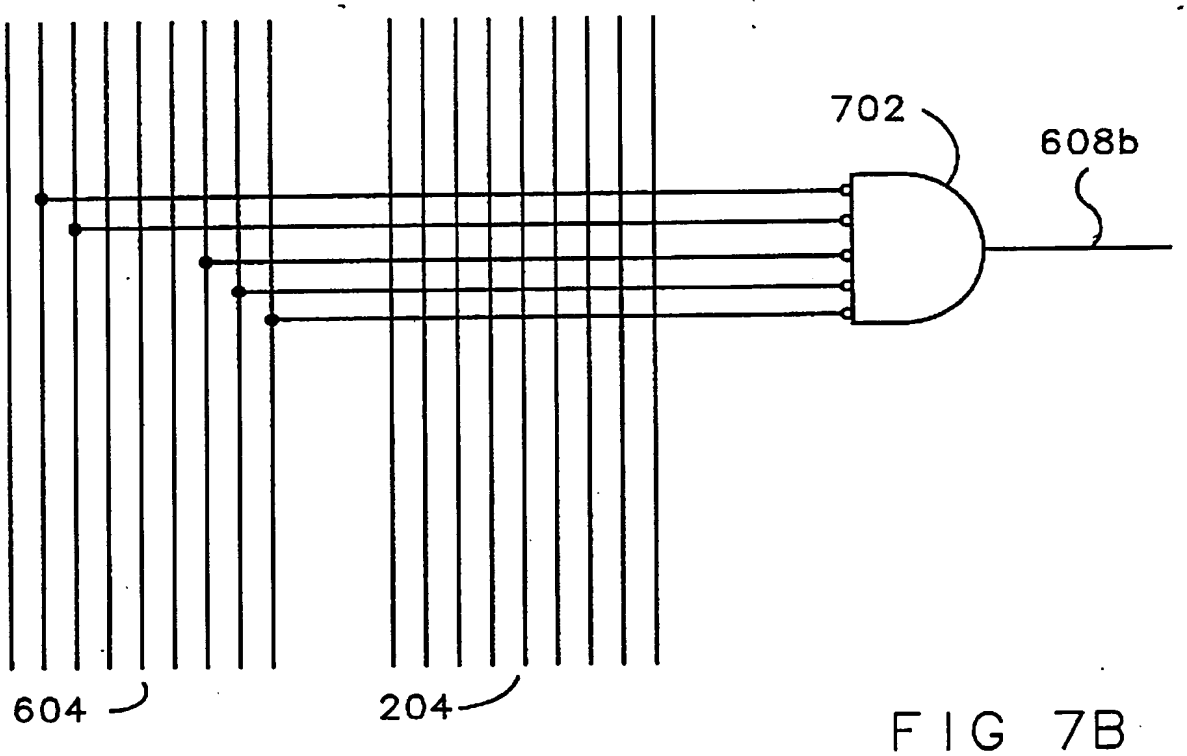

FIG 7B

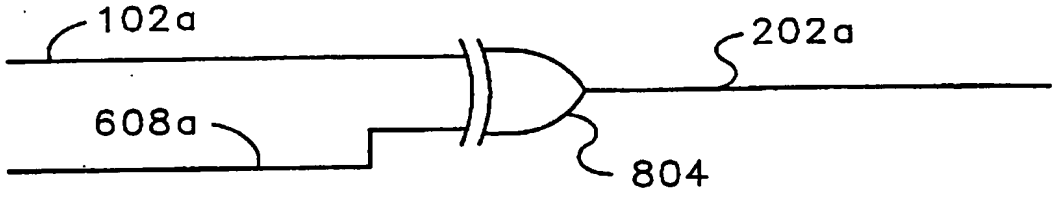

FIG 8

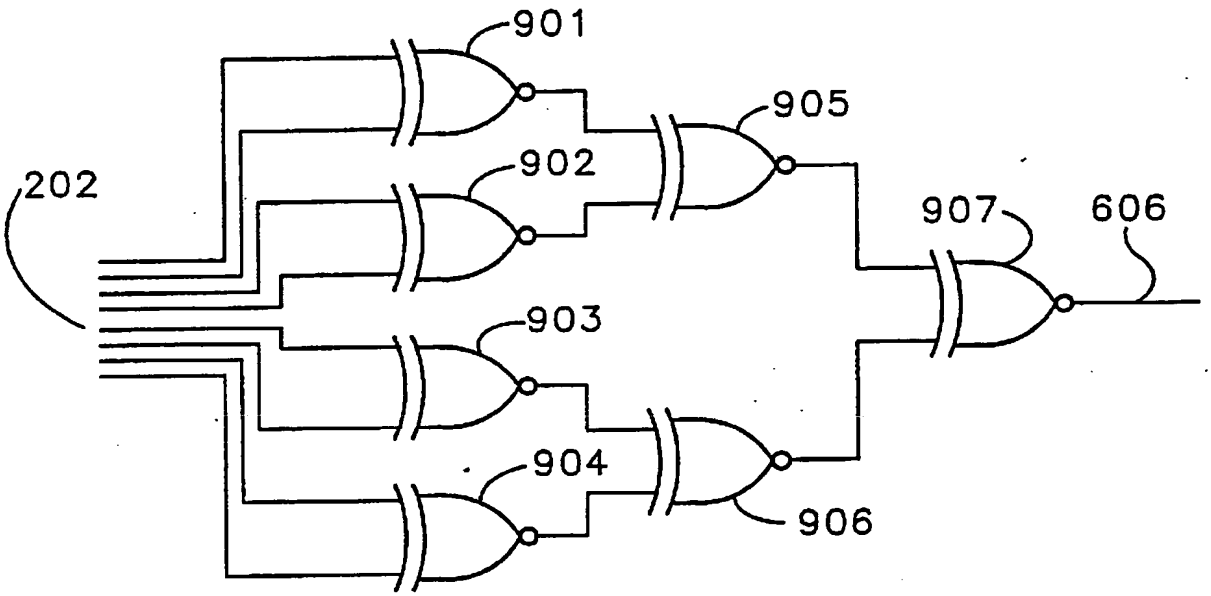

FIG 9A

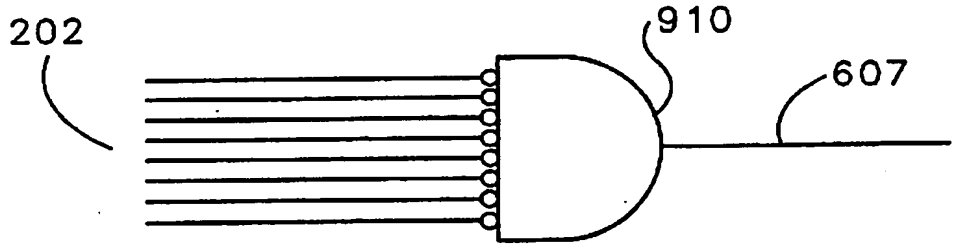

FIG 9B

TABLE 2

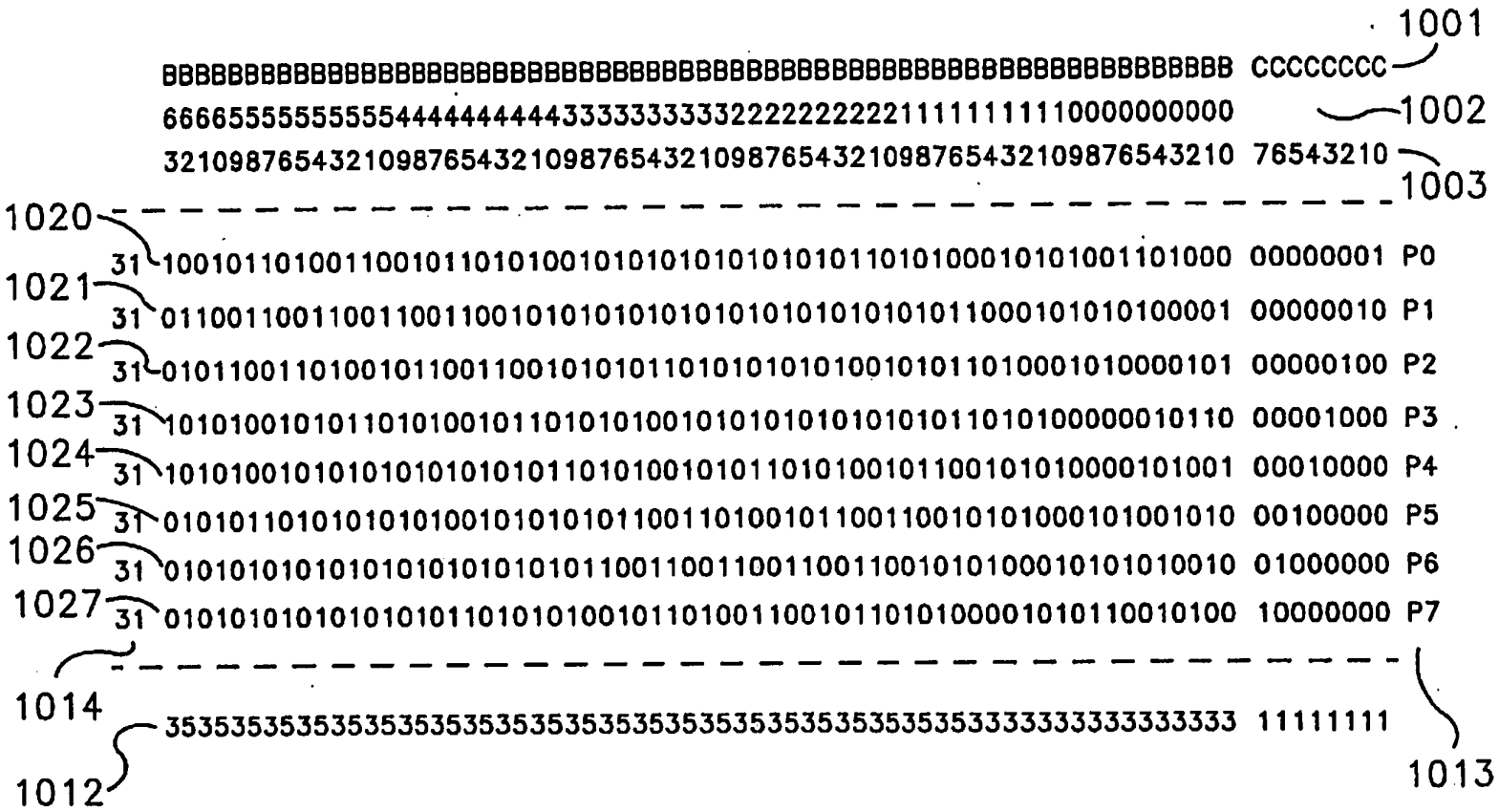

FIG 10

EP 0 310 220 A2